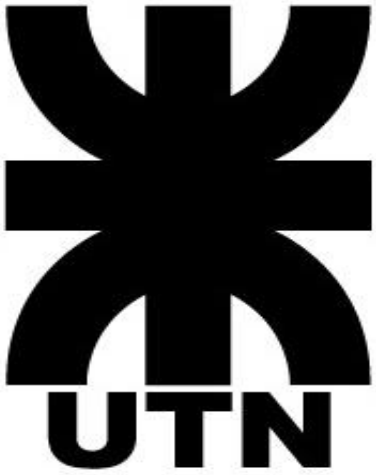

Universidad Tecnológica Nacional
Facultad Regional Concepción del Uruguay

Ingeniería en Sistemas de Información

Sistemas Operativos

Trabajo Práctico Final:
"Ampliación y mejora del shell experimental"

Integrantes del Grupo:

Arrúa, Martin
Lazbal, David
Rivera, Ramiro

Fecha de Entrega:
19/02/2015

Docentes:

Ing. Arellano Gabriel
Ing. Aguiar Osvaldo

```
UNIT comandos;

INTERFACE

Uses
      BaseUnix, Unix, utilidades, users, ALR;

procedure cat     (var dir1,dir2: string; tipo: byte);
           // CAT - Concatena hasta dos archivos, o un archivo con la salida
estandar
procedure exec    (param1: String; param2: Array of AnsiString; tipo:byte);  //
EXEC - Ejecuta un programa externo. Ruta relativa o absoluta.
procedure kill    (signal, proc: longint);
                  // KILL - Envía una señal a un proceso.
procedure ls      (modoA,modoF,modoL:boolean);
           // LS - Lista los archivos de un determinado directorio.

procedure pwd     (tipo: byte);
                        // PWD - Muestra el directorio actual de trabajo.
procedure bg      (ENTRADA:string);
                  // BG - Envia trabajo a segundo plano
procedure fg      (ENTRADA:string);
                  // FG - Envia trabajo al primer plano
procedure jobs;
                              // JOBS - Muestra tabla de trabajos
procedure help;
                              // HELP - Muestra una pantalla de ayuda
procedure moo     (entrada:string);
                  // MOO - Linux users knows ;)


IMPLEMENTATION


//Comando CAT
procedure cat(var dir1,dir2: string; tipo: byte);
var f1,f2: text;
      texto: string;
Begin
      {$I-}       // Evita generar código de control de entrada/salida en el
programa
      assign(f1,dir1);
      reset(f1);
      if IOResult <> 0 then
            begin
            Mostrarerror(10);
            exit;
            end
      else
            begin
            while not eof(f1) do
            begin
                  readln(f1,texto);
                  if tipo = 0 then writeln(texto) else writeln(stdOutPut,texto);
            end;
            close(f1);
            end;
      if dir2 <> '' then
      begin
```

```
            assign(f2,dir2);
            reset(f2);
            if IOResult <> 0 then
                  begin
                  Mostrarerror(10);
                  exit;
                  end
            else
                  begin
                  while not eof(f2) do
                        begin
                              readln(f2,texto);
                              if tipo = 0 then writeln(texto) else
writeln(stdOutPut,texto);
                        end;
                        close(f2);
                  end;
      end;
      {$I+}       // Habilita la generación de código de entrada/salida
End;


//Comando CD
{
      El comando CD es ejecutado directamente
      por el analizadorCD en la UNIT analizador
}

//Comando EXEC
procedure exec (param1: String; param2: Array of AnsiString; tipo:byte);
var pidP,op: longint; proceso:t_procesos;

Begin
      op:= 0;
      pidP:= fpfork;
            case pidP of
            -1: begin
                  Mostrarerror(3);
                  exit;
                  end;
            0:    Begin
                        fpExecLE(param1,param2,envp);
                  end;
            else
                  begin
                        with proceso do
                              begin
                              numero:=TablaJobs.Indice;
                              nombre:=nombreComandoDesdeRuta(param1);
                              pid:=pidP;
                              estado:='Corriendo';
                              prioridad:=' ';
                              directorio:=param1;
                              end;
                        if alBG then
                              begin
                              proceso.estado:=proceso.estado+' &';
                              insertarEnTabla(proceso);
```

```
                                fpWaitPid(pidP,op,WNOHANG); //No "Espera" por el
hijo con pid=PidP
                                end
                          else
                                begin
                                pidenejec:=pidP;        // Cargo el pid del nuevo
programa como en ejecucion actual
                                fpWaitPid(pidP,op,0); //Espera por el hijo con
pid=PidP
                                if not procesoFinalizado(op) then
                                      insertarEnTabla(proceso);
                                pidenejec:=-1;          //borro el pid asignado an-
tes
                                end;
                    end;
              end;
End;


//Comando KILL
procedure kill (signal, proc: longint);
Begin
      fpKill(proc,signal);
End;


//Comando LS
procedure ls (modoA,modoF,modoL:boolean);
{
      Segun los parametros booleanos distingue entre dos tipos de LS
      y dentro de los mismos actua acorde a dichas variables
}
begin
if modoL then
      lsL(modoA,modoF,modoL)
else
      lsAF(modoA,modoF);
end;


procedure lsAF(modoA,modoF:boolean);
{
      Los archivos marcados como inaccesibles son aquellos
      en los cuales fpStat devuelve un error. En dicho caso se puede saber
      el nombre del archivo, pero no datos referidos al tipo de archivo.

}

var   directorio: pdir;
      entrada: PDirent;
      vector: vDirent;
      indice,j: integer;
      K: byte;
      info: stat;
      auxNombre,tipo    : string;

Begin
      k:= 1;tipo:='';
      indice:= 0;
```

```
      directorio := fpOpenDir(dirActual);
      if directorio <> nil  then
      begin //openDir
      repeat
            entrada:= fpReadDir(directorio^);
            if (entrada <> nil) and (modoA or (entrada^.d_name[0] <> '.')) then
//Si  mostrarTodos (ModoA) o no Oculto, lo carga al vector de directorio
                  begin
                  inc(indice);
                  vector[indice]:= entrada^;
                  end;
      until entrada = nil;
      if modoA then // si ModoA ordena el directorio alfabeticamente
            burbujaDirent(vector,indice);
      for J:= 1 to indice do
            Begin //for
            if fpStat(pchar(vector[J].d_name),info)=0 then
                  Begin  //fpStat
                  if modoA then
                        tipo:=tipoArchivoAF(info.st_mode)+' ';
                  auxNombre:=tipo+copy(vector[J].d_name,1,20);
                  case K of //case
                  1:    begin
                              if j <> indice then
                                    write(auxNombre+espacio(24-length(auxNom-
bre)))
                              else
                                    writeln(auxNombre)
                        end;
                  2:    begin
                              if j <> indice then
                                    write(auxNombre+espacio(24-length(auxNom-
bre)))
                              else
                                    writeln(auxNombre)
                        end;
                  3:    begin
                              writeln(auxNombre+espacio(24-length(auxNombre)))
                        end;
                  end; // case
                  if k = 3 then k:= 1 else inc(K);
            End // fpStat
            else
            Begin // else fpStat
                  if modoA then
                        tipo:=tipoArchivoAF(info.st_mode)+' ';
                  auxNombre:=tipo+copy(vector[J].d_name,1,20);
                  case K of //case
                  1:    begin
                              if j <> indice then
                                    write('Inaccesible: '+auxNombre+espacio(24-
length(auxNombre)))
                              else
                                    writeln('Inaccesible: '+auxNombre);
                        end;
                  2:    begin
                              if j <> indice then
                                    write('Inaccesible: '+auxNombre+espacio(24-
length(auxNombre)))
```

```
                              else
                                    writeln('Inaccesible: '+auxNombre)
                        end;
                  3:    begin
                              writeln('Inaccesible: '+auxNombre+espacio(24-len-
gth(auxNombre)))
                        end;
                  end; //case
                  if k = 3 then k:= 1 else inc(K);
            end; // else fpStat
      End; // for
      fpCloseDir(directorio^);
      end  // OpenDir
      else
            begin
            mostrarerror(4);  exit;
            end;
End;



//comando LS -l
procedure lsL(modoA,modoF,modoL:boolean);
{
      En caso de falla obteniendo los datos de un archivo se mostrará
      el mensaje "No se pudo mostrar.", que representa una falla en fpStat.

}

var   directorio  : Pdir;
      entrada           : PDirent;
      info        : stat;
      vector            : vDirent;
      contArchivos,loop: integer;
      indice,j    : integer;
      tamString,unidad  : string;
      tamanio: int64;

Begin
      indice:= 0;
      contArchivos:= 0;       //número de archivos listados
      directorio:= fpOpenDir(dirActual);
      if directorio <> nil then
      begin //openDir
            repeat
                  entrada:= fpReadDir(directorio^);
                  if (entrada <> nil) and ((modoA) or (entrada^.d_name[0] <>
'.')) then//Si  mostrarTodos (ModoA) o no Oculto, lo carga al vector de directo-
rio
                  Begin
                        inc(indice);
                        vector[indice]:= entrada^;
                  end;
            until entrada = nil;
            if modoA then // si ModoA ordena el directorio alfabeticamente
                  burbujaDirent(vector, indice);
            for J:= 1 to indice do
                  begin //for
                  if fpStat(pchar(vector[J].d_name),info)=0 then
```

```
                        Begin //fpStat
                              contArchivos:= contArchivos + 1;
                              tamanio:=info.st_size;
                              write(modoACadena(info.st_mode));
                              write(permisosACadena(pchar(vector[J].d_name)));
                              write(tamAsString(info.st_nlink,2),espacio(1));
                              write(getusername(info.st_uid)+espacio(2));
                              write(getgroupname(info.st_gid)+espacio(1));

                              loop:=1;
                              if (tamanio div 1024) > 1000 then
                                    begin
                                    repeat
                                          if (tamanio div 1024) > 1000 then
                                                begin
                                                tamanio:=tamanio div 1024;
                                                end;
                                          inc(loop);
                                    until tamanio < 100000;
                                    end;
                              case loop of
                                    1:unidad:=' bytes';
                                    2:unidad:='Kbytes';
                                    3:unidad:='Mbytes';
                                    4:unidad:='Gbytes';
                                    end;

                              tamString:=' '+tamAsString(tamanio,6)+unidad;
                              write(tamString+espacio(15-length(tamString)));
                              write(tipoArchivoL(info.st_mode)+espacio(2));
                              write(tiempoUnixAHumano(info.st_mtime)+espa-
cio(2));
                                    writeln(pchar(vector[J].d_name));
                        End // fpStat
                        else
                              writeln('No se pudo mostrar.');
                  end; //for
                  writeln('- - - - - - - - - - - - - - - - -');
            if contArchivos = 0 then
                  writeln('No hay listado.')
            else
                  writeln('Nro. de archivos listados: ',contArchivos);
            fpCloseDir(directorio^);
      end //openDir
      else begin
            mostrarerror(4);  exit;
            end;
End;



//Comando PWD
procedure pwd(tipo: byte);
var pid, op: longint;
Begin
      pid:=fpFork;
      op:= 0;
      case pid of
      -1:begin
```

```
            mostrarerror(3);  exit;
            end;
       0:   begin
                  writeln(dirActual);
                  fpKill(fpGetPid,9);
            end;
      else
            begin
                  fpWaitPid(pid,op,op); //Espera por cualquier proceso hijo.
            end;
      end;
End;

procedure bg (ENTRADA:string);
var
      numeroTrabajo: longint;
      pid,error:longint;
begin
      val(ENTRADA,numeroTrabajo, error);
      pid:=damePid(numeroTrabajo);
      if pid = -1 then
            writeln('Proceso no Encontrado')
      else
            begin
                  fpkill(pid,SIGCONT);
            end;
end;

procedure fg (ENTRADA:string);
var
      numeroTrabajo: longint;
      pid,error,pid2:longint;
      estado:longint;
begin
      val(ENTRADA,numeroTrabajo, error);  //Entrada es SIEMPRE un numero si
quiere funcionar (numero en String)
      pid:=damePid(numeroTrabajo);  //Obtengo el pid correspondiente al numero
de trabajo
      if pid = -1 then // No se encontro numero de trabajo
            begin
            mostrarError(20);
            exit;
            end
      else
            begin
            pidEnEjec:=pid; //cargo el programa que se va a traer al FG
            pid2:=fpfork;     // necesito que la terminal "duerma" mientras hay
otra cosa en FG
            case pid2 of
                  0: begin
                        FpSetsid;   { Creo una nueva sesion para el proceso que
debe ser traido al frente
                                                http://www.freepascal.org/docs-
html/rtl/baseunix/fpsetsid.html
                                                http://stackoverflow.com/ques-
tions/9306100/how-can-i-tell-if-a-child-is-asking-for-stdin-how-do-i-tell-it-to-
stop-that}
                        fpkill(pid,SIGCONT); //Envio senial de resumen al pro-
ceso a FG
```

```
                        end;
                    else
                        begin
                        fpwaitpid(pid,estado,0); //Espero que el hijo(proceso
actualmente en FG) se detenga
                        fpkill(SIGKILL,pid2);
                        ActualizarEstado(pid,estado); //actualizo el estado del
proceso que habia enviado a FG
                        pidenejec:=-1; // borro el id del proceso que se estaba
ejecutando
                        end;
                    end;
                end;
end;


procedure jobs;
begin
      mostrarTabla;
end;



procedure help;
begin
      writeln('ALROShell');
      writeln();
      writeln('Copyright 2015');
      writeln('* Ramiro Rivera                    <ramarivera@gmail.com>,');
      writeln('* David lazbal                  <davidlazbal@gmail.com>,');
      writeln('* Matín Arrúa                   <martin94.profugo@gmail.com>.');
      writeln('Copyright 2014');
      writeln('* Fernando Gómez Albornoz       <fgalbornoz07@gmail.com>. ');
      writeln();
      writeln('Estas órdenes del shell están definidas internamente');
      writeln();
      writeln('bg id_trabajo');
      writeln('cat primer_archivo [segundo_archivo]');
      writeln('cd ruta');
      writeln('exit');
      writeln('fg id_trabajo');
      writeln('jobs');
      writeln('kill -id_señal id_proceso');
      writeln('ls {[-(l|f|a)]} (3 Parametros máximo) [ruta]');
      writeln('pwd');
      writeln();
      writeln('Este shell acepta los siguientes operadores: ">" ">>" "&" "|"');
      writeln();
      writeln('Para más información diríjase a la página man');
end;

procedure moo(entrada:string);
var
      aux:string;
      I:longint;
begin
      if upcase(ENTRADA)<>'-H' then
            begin
            if sinEspacios(Entrada)=''  then
```

```
                aux:='Si yo estoy aca es porque CRT no lo esta :)'
            else
                aux:=entrada;
            write(' ');
            for I:=1 to length(aux)+2 do
                write('_');
            write(CR+LF);
            writeln('( '+aux+' )');
            write(' ');
            for I:=1 to length(aux)+2 do
                write('-');
            writeln(' ');
            writeln('        o   ^__^');
            writeln('         o  (oo)\_______');
            writeln('            (__)\       )\/\');
            writeln('                ||----w |');
            writeln('                ||     ||');
            writeln();
            end
        else
            begin
            writeln(space(30)+'Historia de Moo y los Super Cow Powers');
            writeln('Very well internet, you win. Let me tell you a tale about
cow powers. Super ones to be specific.');
            writeln('Once a long time ago a developer was known for announcing
his presence on IRC with a simple, to the point '+chr(39)+'Moo'+chr(39)+'.');
            writeln('As with cows in pasture others would often Moo back in
greeting. This led to a certain range of cow based jokes.');
            writeln('When apt-get was initially developed I put the enigmatic
tag line in the help message, but I did not add the '+chr(39)+'apt-get
moo'+chr(39)+' command.');
            writeln('That act lies with another, who decided that the help
teaser needed some extra zip. Thus the easter egg was born.');
            writeln('The items in aptitude are probably a homage, as aptitude
was substantially based on apt'+chr(39)+'s library.');
            writeln('It seems very popular, it was featured in Linux Magazine
some time ago, and I'+chr(39)+'ve even had people request a Moo when they find
me at conferences.');
            writeln('There have been bug reports to remove it, explain it, and
to improve the cow.');
            writeln('It was mentioned for a while  in Wikipedia, and now appar-
ently on stack exchange.');
            writeln('Now, if you look closely, in the right places, you can find
other software with cow powers. Good luck :)');
            writeln(space(50)+'-Jason Gunthorpe');
            writeln();
            end;
end;

END.
```